Estimativa de Esforço de Software Usando Raciocínio Baseado em Casos

Iris Fabiana de Barcelos Tronto, Nilson Sant'Anna, José Demísio Simões da Silva
Ministério da Ciência e Tecnologia – MCT
Instituto Nacional de Pesquisas Espaciais – INPE
Laboratório Associado de Computação e Matemática Aplicada – LAC/CAP
E-mail:(iris_barcelos,nilson,demísio)@lac.inpe.br

Resumo

A gestão de projeto de software é uma atividade em que os gerentes usam suas habilidades e experiências para tomar decisões. A habilidade de se realizar predições efetivas, particularmente no que se refere a custo e qualidade, é uma das maiores contribuições para se ter um gerenciamento de projeto de software de sucesso. Conseqüentemente, pode-se verificar a existência de atividades de pesquisa significativas nesta área, muitas das quais têm o foco sobre predição de esforço e de defeitos. Estes problemas são caracterizados por ausência de teoria, inconsistência e incerteza que fazem deles bem ajustados a abordagens de raciocínio baseado em casos - RBC. Este trabalho apresenta um estudo de caso que aplica RBC para realizar estimativa de esforço de desenvolvimento de componentes de software.

Palavras-chave: *gestão de projetos de software, estimativas de software, mineração de dados, RBC.*

1. Introdução

Nos últimos anos têm sido relatadas várias histórias de insucesso no desenvolvimento de software através de estudos de casos e experimentos documentados. Uma grande parte dos projetos de desenvolvimento ainda usa mais recursos do que aqueles que foram planejados, eles levam mais tempo para serem concluídos, fornecem menos funcionalidade e menos qualidade que o esperado. Muitos estudos relacionam tais falhas ao gerenciamento de projeto inadequado, apontando problemas de comunicação, alocação errada de habilidade da equipe, treinamento insuficiente e falta de habilidade de gerentes para predizer e ajustar o comportamento do projeto. Além disso, projetos são caracterizados por incertezas.

Mesmo diante dessas incertezas gerentes tendem a aplicar algumas técnicas tradicionais, que trabalham bem na teoria, mas que não podem ser aplicadas diretamente na prática. Entretanto, gerentes experientes são capazes de perceber alguns aspectos de um processo de desenvolvimento de software que um gerente novato não percebe. Este conhecimento gerencial contido na base mental de um gerente especialista é um recurso de grande valor. Portanto, é importante utilizar técnicas para representar e proporcionar uma reutilização deste conhecimento, de forma que um gerente novato possa compartilhar as experiências dos especialistas.

Este conhecimento é registrado a partir de um programa de medição implantado na organização e armazenado em um repositório de experiência [5], que aumenta à medida que projetos vão sendo terminados. Exemplos de medidas são: densidade de defeitos, tamanho da aplicação, quantidade de esforço, custos e medidas relacionadas ao cronograma, dentre muitas outras. Assim, uma grande quantidade de dados é armazenada e muita informação útil pode permanecer oculta nesses repositórios de experiência, sendo desejável que se apliquem técnicas mais sofisticadas de exploração dos dados para auxiliar gerentes na tomada de decisões no planejamento e acompanhamento de projetos.

Várias pesquisas têm sido conduzidas com o objetivo de construir, avaliar e recomendar técnicas para análise exploratória de dados de projetos. Dentre elas, técnicas estatísticas e de Inteligência Artificial. Mendonça e Sunderhaft relatam a aplicação de técnicas de mineração de dados como uma ferramenta útil para explorar dados de engenharia de software [10], [11]. OLAP (Processamento Analítico – *On-Line Analytical Processing)* é uma tecnologia que também pode ajudar gerentes na tomada de decisão através de uma análise multidimensional das informações [4]. Outros trabalhos têm sido realizados, por exemplo, pesquisas utilizando a metodologia de Raciocínio Baseado em Casos (RBC), que apresenta bons resultados tanto para reuso quanto para realizar estimativas de projeto [9], [15].

O objetivo desse trabalho é apresentar um estudo de caso em que se aplica RBC para predizer o esforço de projeto de software.

Esse trabalho está estruturado em cinco capítulos, sendo o Capítulo 1 esta Introdução. O Capítulo 2 apresenta alguns trabalhos relacionados à aplicação de técnicas de analise exploratória de dados para realizar predição. O Capítulo 3 apresenta a metodologia de RBC. O Capítulo 4 descreve um estudo de caso, utilizando a metodologia de RBC para representar o conhecimento e realizar a predição de esforço, utilizando dados de projetos de software. O Capítulo 5 apresenta as conclusões do trabalho. E, por fim, as referências bibliográficas.

2. Trabalhos relacionados

Há uma tendência por se usar técnicas de mineração de dados para análise exploratória de dados de engenharia de software, conforme pode ser observado no último *Workshop Internacional sobre Mineração de Repositório de Software* [12]. Mineração de dados, no contexto de engenharia de software, é definida como um processo de indução de informação potencialmente útil e previamente desconhecida de bases de dados de desenvolvimento de software [10].

Dentre outras técnicas para a análise exploratória de dados, modelos probabilísticos, *clustering* e Raciocínio Baseado em Casos têm se mostrado eficiente para realizar predições. Em [18] *clustering* é aplicada para realizar estimativas de qualidade de software. Uma abordagem probabilística é utilizada em [16] para predição de custo. Há uma grande tendência de se aplicar RBC para resolver problemas relacionados ao planejamento e acompanhamento de projetos de software, que podem ser classificados em duas categorias: aplicações do tipo reuso e do tipo predição.

Boehn, no início da década de 80, sugeriu que analogia formava uma boa base para predição de esforço de software [7]. Vicinanza e seus colaboradores desenvolveram um sistema CBR, denominado *Estor*, que era comparável a um especialista e significativamente mais preciso que o modelo para estimativas COCOMO [7] ou Pontos por Função [2]. No entanto, sua abordagem requer acesso a um especialista, de forma a derivar as regras de estimativas e criar uma Base de Casos. FACE (Encontrando Analogias para Estimativas de Custo – *Finding Analogies for Cost Estimatio*) é outro projeto que também usa a tecnologia RBC [6]. Posteriormente, foi desenvolvida uma abordagem aparentemente melhor que outras baseadas em regressão e comparável àquelas que utilizam redes neurais (ANN – *Artificial Neural Net*) [8].

RBC apresenta algumas vantagens em relação a outras técnicas de gerenciamento de conhecimento e paradigmas da Inteligência Artificial para se realizar predição [1]. Dentre elas: a) evita muitos dos problemas associados com especificação e codificação do conhecimento; b) manipula casos que falharam, os quais permitem aos usuários identificar situações propensas a alto risco; c) usuários freqüentemente estão mais dispostos a aceitar soluções de sistemas baseados em analogia por esta ser uma forma semelhante àquela com que um humano resolve um problema.

3. Raciocínio Baseado em Casos

RBC é uma técnica de resolução de problemas por analogia, que utiliza a experiência para resolver novos problemas. O raciocínio por analogia consiste na identificação de aspectos similares em problemas anteriores e atuais com o intuito de avaliar a utilização das soluções encontradas para a formulação de uma nova solução. Para Aamodt e Plaza, sistemas RBC apresentam a capacidade de utilizar o conhecimento referente a experiências passadas (casos) para resolver novos problemas, sendo que a solução é encontrada através da recuperação de um caso similar do passado. Casos são caracterizados por vetores de características tais como o tamanho de um arquivo, o número de interfaces ou o método de desenvolvimento. Sistema de RBC resolve um novo problema (relacionado ao *novo caso*) por meio da recuperação e adaptação de casos similares de um repositório de casos passados (e portanto resolvidos). O repositório é denominado *Base de Casos* (BC).

Aamodt e Plaza [1] identificaram quatro estágios de RBC (às vezes, referido como o modelo R^4): codificação, recuperação, reuso e retenção. Eles são combinados para formar um processo cíclico, conforme ilustrado pela Figura 1.

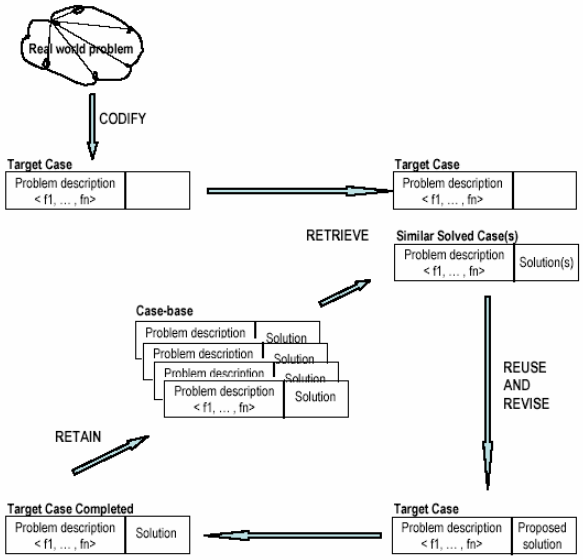

FIGURA 1 Processo de raciocínio baseado em casos
FONTE: adaptada de [1]

A Base de Casos está no centro, que é um repositório de casos completos, em outras palavras, é a memória. Quando um novo problema surge, ele deve ser codificado em termos do vetor de características (ou descrições do problema) que é então a base para se recuperar casos similares de Base de Casos. Claramente, quanto maior o grau de sobreposição de características, mais efetivas serão as *medidas de similaridade* e a recuperação do caso. Independentemente da medida de similaridade utilizada, o objetivo é classificar os casos em ordem decrescente de similaridade ao *novo caso* e utilizar as soluções conhecidas dos *k* casos mais próximos. Soluções derivadas dos casos recuperados podem ser adaptadas para melhor ajustar o *novo caso* por meio de regras, de um especialista humano ou por um simples procedimento estatístico. Para este último caso, o sistema é chamado como Técnica do Vizinho Mais Próximo *(K-Nearest Neighbour – K-NN*). Quando o *novo caso* tiver sido completado e a solução

verdadeira for conhecida, ele pode ser retido na Base de Casos. Dessa forma, a Base de Casos cresce com o tempo e o novo conhecimento é adicionado. É claro que é importante não descuidar da manutenção da Base de Casos de forma a prevenir a degradação em relevância e consistência. Alguns conceitos importantes que compõem este ciclo são ilustrados pelo estudo de caso mostrado no Capítulo 4.

4. Estimativa de esforço usando RBC

Aqui é apresentado um estudo de caso, que consiste da aplicação da metodologia de raciocínio baseado em casos para extrair informações da base de dados denominada SEL, de forma que possam auxiliar um gerente de projetos a predizer a quantidade de recursos que deverá alocar para o desenvolvimento de um novo componente de software. O objetivo é realizar a estimativa da quantidade de horas gastas com programação e a quantidade de horas gastas com gerenciamento, baseado em experiências passadas. Essa base foi obtida de um Laboratório de Engenharia de Software (*Software Engineering Laboratory* – SEL), que a disponibilizou gratuitamente em sua página. Ela contém dados relacionados à gestão de componentes de software, os quais são de diferentes tipos, desenvolvidos em diferentes linguagens de programação e para realizar diferentes funções.

4.1. Planejamento

Nesta fase foi feita uma análise sobre possíveis técnicas, algoritmos e ferramentas que poderiam ser utilizados na representação, armazenamento e recuperação dos casos em uma Base de Casos. Existem várias técnicas de representação de conhecimento que podem ser utilizadas pela metodologia RBC, dentre elas quadros, árvores semânticas e ontologias. Aqui se optou por utilizar quadros. Utilizou-se o Microsof Excell para modelar os dados dos projetos (Diagrama Entidade e Relacionamento - DER) e para representar os quadros (*frames*). Este aplicativo foi escolhido devido à facilidade de usar e por atender aos requisitos e às necessidades deste estudo. Utilizou-se o algoritmo K-Vizinhos mais próximos (k_NN) na busca pelos casos mais similares ao novo problema. A escolha do algoritmo de similaridade na fase de planejamento é importante, já que a preparação dos dados depende do algoritmo escolhido. Por exemplo, para que o algoritmo k-NN produza resultados mais confiáveis, é aconselhável considerar que todos os valores contidos nas características dos casos sejam do tipo numérico e estejam normalizados.

O ambiente Visual Prolog foi utilizado para implementar os algoritmos. Prolog é uma linguagem de programação de Inteligência Artificial, que permite implementar os conceitos de Raciocínio Baseado em Casos de uma forma simples e natural. Além disso, o ambiente Visual Prolog possui uma interface gráfica "amigável".

4.2. Análise e entendimento da base de dados

A base foi obtida da página de um laboratório de engenharia de software, que não disponibilizou uma documentação adequada. Isso dificultou a compreensão do relacionamento entre os dados e a identificação das características que se julgam interessantes para realizar a estimativa de esforço.

Após esta análise, identificou-se um conjunto inicial de características – utilizando conceitos obtidos previamente sobre gestão de projetos – que poderiam compor os casos. Ao confrontar as características básicas com os atributos das tabelas na base de dados, verificou-se que não existia um relacionamento entre muitas dessas características e os atributos na base de dados, o que contribuiu para criação de outros conjuntos de características., até que um último conjunto foi escolhido. Não seguiu nenhum procedimento sistemático para a escolha das características.

Com base neste último conjunto de características, construiu-se um Diagrama de Entidade e Relacionamento (DER), para identificar a relação entre estas características na base de dados.

Esse conjunto de características básicas é constituído por aquelas utilizadas na busca, pelas características para as quais se deseja estimar o valor e por outras que o usuário deseje visualizar. Nesse estudo de caso, por limitação de tempo, restringiu-se a implementação a: 1) características utilizadas na busca: *Modified_Component,New_Componente, Total_Line, Programmer_HR, Reused_Component, Total_Component, Total_Mode_Line, Total_New_Line, Total_Old_Line, Total_Subsystems;* 2) características para as quais se deseja estimar o valor: *Management_HR, Programmer_HR (*que são horas despendidas com gerenciamento, horas despendidas com programação).

4.3. Representação do conhecimento

Nesta etapa fez-se a representação do conhecimento em quadros (*frames*). O DER foi utilizado para verificar o relacionamento entre os dados contidos nas tabelas relacionadas com o conjunto de características identificado e assim construir os quadros.

O quadro "Área de Atuação" significa a definição da área para a qual a estimativa será realizada. Por exemplo, uma área poderia ser gerenciamento de projetos, programação, dentre outras. Para este estudo de caso, a área de atuação escolhida foi gerenciamento de projetos. Já o quadro "Categoria de Projeto" significa a definição da

categoria a que o projeto pertence. Por exemplo, uma categoria poderia ser gerenciamento de projeto de software, outra poderia ser gerenciamento de projeto de hardware. Para este estudo de caso foi considerada somente a categoria gerenciamento de projeto de software. O quadro "Form" está relacionado a um projeto, para o qual são realizadas estimativas. O quadro "Estimar_projeto" apresenta as características de projeto para os quais estimativas são realizadas. Um projeto, representado pelo quadro "Projeto", está relacionado a uma aplicação. Esta aplicação, representada pelo quadro "Programa de Aplicação", é constituída de vários componentes. Um componente é representado pelo quadro "Componente".

4.4. Transformação dos dados

Esta seção descreve as transformações que foram feitas na base de dados SEL, a fim de ajustá-la às necessidades impostas pelos algoritmos utilizados na metodologia RBC.

Redução de campos de dados inúteis: em algumas tabelas da base de dados, existem campos que não têm utilidade para auxiliar no planejamento de um outro projeto, por não possuir dados em nenhum dos registros da tabela ou por não estar no contexto da análise em questão. Por exemplo, na tabela PROJECT, há quatro campos (APPLICATION, ORGANIZATION, TYPE, DEV_END_D) que são explicitamente inúteis por não possuir dado em nenhum dos registros da tabela. Neste caso, os quatro campos foram excluídos. O mesmo procedimento foi realizado para as outras tabelas utilizadas no estudo de caso.

Tratamento de campos vazios: conjunto de dados SEL contém uma grande quantidade de dados perdidos – ou seja, campos vazios nas tabelas – o que torna a predição difícil. O problema de valores de dados perdidos tem sido estudado por muitos pesquisadores e soluções comuns para esse problema incluem técnicas de omissão [14], técnicas de imputação [17] *e técnicas de tolerância de dados perdidos* [3]. *As técnicas de omissão simplesmente excluem os casos que contêm dados perdidos. Por causa desta simplicidade, elas são amplamente usadas, mas, por outro lado, podem levar a um mau uso dos dados se há uma grande quantidade de dados perdidos no conjunto de dados. As técnicas de tolerância são as estratégias de tratamento interno dos dados perdidos, que fazem análise utilizando o conjunto de dados com valores perdidos. As técnicas de imputação estimam valores para os casos com valores perdidos e troca os valores perdidos por estimativas, que são obtidas baseando-se em valores relatados e produz um caso completo estimado.* Nesse estudo de caso, foi utilizada a técnica de omissão. Todos os registros que continham dados perdidos no campo correspondente a qualquer característica de busca (Modified_Component,New_Componente, Programmer_HR, Reused_Component, Total_Component, Total_Line, Total_Mode_Line, Total_New_Line, Total_Old_Line, Total_Subsystems) foram excluídos. Com isso, houve uma redução aproximada de 50% dos casos. Esta técnica foi empregada a fim de facilitar o cálculo da distância entre dois pontos e por ser a técnica mais simples e mais amplamente usada.

Tratamento de dados simbólicos: os dados podem ser de diferentes tipos (por exemplo, numéricos, nominais ou simbólicos, etc). Nesse estudo de caso, algumas das características são do tipo nominal, por exemplo, a característica Função. Para obter resultados mais precisos da aplicação do algoritmo de similaridade k_NN – utilizado nesse trabalho para cálculo de similaridade entre casos – é preciso transformar esses dados simbólicos em dados numéricos. A estratégia adotada foi duplicar cada coluna que contém dados simbólicos, colocando os valores convertidos na coluna correspondente (duplicada). Para a conversão, sugere-se utilizar seis dígitos binários para representar cada valor simbólico diferente encontrado nos registros da base de dados. Foi feita a transformação dos dados simbólicos da coluna FUNCTION, da Tabela APPLICATION_PROGRAM em dados numéricos e colocados na coluna FUNCTION NUMERIC. Por exemplo, o dado DATA PROCESSING/DATA CONVERSION da coluna FUNCTION possui o valor numérico correspondente igual a 100000, armazenado na coluna Function Numeric.

Normalização do conjunto de dados: ao aplicar o algoritmo de similaridade é calculada a distância euclidiana entre as características do caso analisado e as características do novo caso. Como resultado deste cálculo podem-se ter valores muito diferentes. Nesse caso, uma estratégia é normalizar os dados de forma que os valores que representam as distâncias entre características estejam em um intervalo entre zero e um. O primeiro passo é estabelecer um valor máximo e um valor mínimo que cada característica pode assumir. Em seguida, para calcular o valor normalizado de cada característica para todos os casos da base, aplica a fórmula:

$$\text{valor normalizado} = \frac{(\text{valor máximo} - \text{valor da característica})}{(\text{valor máximo} - \text{valor mínimo})}$$

4.5. Criação de uma base de casos

O conhecimento extraído da Base de Experiência é representado em forma de casos em uma memória para uso do Prolog. Estes casos são representados em forma de fatos e formam a Base de Casos. Esta extração de conhecimento foi feita através do comando SQL "tsql_Exec". Os dados da visão criada são recuperados para a memória utilizada

no Prolog através de *retrieve_dataset*: Cada caso será constituído de um vetor de características para descrever cada componente. Exemplos de características devem incluir: *Modified_Component,New_Componente, Programmer_HR, Reused_Component, Total_Component, Total_Line, Total_Mode_Line, Total_New_Line, Total_Old_Line, Total_Subsystems.*

4.6. Execução de estimativas para um novo caso

Quando um novo problema de predição surgir, o novo componente deve ser descrito em termos do vetor de características de modo que ele possa ser visto como o *novo caso*. Assim, as características importantes para se realizar a busca são preenchidas com seus respectivos dados e as características, para as quais as informações devem ser estimadas são preenchidas com zero. Para o estudo de caso, o problema é recuperar casos similares da Base de Casos, usando os valores: a quantidade de horas despendidas com gerenciamento e a quantidade de horas despendidas com programação. Na recuperação dos casos similares, utilizou-se o algoritmo de similaridade *k*_NN.

O vetor de dados de entrada pode incluir características de tipos diferentes. Isso adiciona alguma complexidade para a forma através da qual a distância é medida. Assim, é preciso transformar os dados de entrada de forma que se possa diminuir tal complexidade. Nesse trabalho, propõe-se transformar os dados simbólicos em dados numéricos e normalizar os dados, conforme realizado para os dados da Base de Experiência.

No algoritmo de busca (denominado *annSearch*) pelo caso mais similar ao novo caso, os dados informados como parâmetros são: *o número 1,* que é o número de casos mais próximos que se deseja recuperar. Neste exemplo, considerou-se **um** caso mais semelhante; *os demais valores numéricos,* que são os dados do novo caso, incluindo os campos a serem estimados (quantidade de horas de programação e horas de gerenciamento, que são inicialmente definidos com o valor zero); *os dados denotados pelas letras A2...A12,* que são os valores mínimos de cada característica; *os dados denotados pelas letras Z2...Z12,* que são os valores máximos de cada característica. Estes dois últimos conjuntos de dados são importantes para permitir a normalização dos dados de entrada do novo caso, a fim de permitir o calculo da distância (e, conseqüentemente, da similaridade) entre os casos e o novo caso.

Neste algoritmo, primeiramente, faz-se a normalização dos dados de entrada. Depois de ter os dados normalizados, calcula-se a diferença entre os dados (normalizados) das características do novo caso e os dados das características de um caso da base. Eleva ao quadrado a diferença do valor de cada característica e a soma destas diferenças representa a distância (similaridade) entre um caso e o novo caso. Esse cálculo é feito para todos os casos da base e aquele que apresentar a menor distância é o mais similar e deve ser apresentado ao usuário como sendo a estimativa solicitada.

5. Conclusão e discussão dos resultados

Pelas várias vantagens da metodologia de Raciocínio Baseado em Casos (RBC), identificadas através desse estudo de caso, viu-se que ela se mostra promissora para realizar estimativas de software. Porém, o nível de confiabilidade dos resultados alcançados para predições foi considerado impróprio, sendo necessário realizar pesquisas no sentido de alcançar resultados mais satisfatórios. Além disso, foi possível experimentar vários problemas relacionados à análise exploratória de dados, utilizando RBC.

Um problema se refere à Base de Dados SEL utilizada para desenvolver esse trabalho e que não possuía uma documentação adequada disponível. Isso dificultou a identificação do conjunto de características relevantes para compor os casos. Por outro lado, essa é uma realidade que se pode esperar de uma Base de Experiência de uma organização de software que não tenha um programa de medição implementado e que esteja em um nível de maturidade baixo.

Uma questão importante é definir a hierarquia na modelagem e determinar quais os atributos que mais interferem para uma determinada estimativa. Na representação do conhecimento feita nesse estudo de caso, foi determinado que o *tipo de projeto* pode ser determinante na otimização da busca por casos similares, de forma que se restringe a busca por tipo de projeto, ou seja, se o projeto é de tempo real, então, a busca é realizada entre os casos cujo tipo de projeto é de tempo real; se o projeto é um sistema científico, então, a busca é realizada entre os casos cujo tipo de projeto é científico e assim segue para todos os tipos possíveis estabelecidos. Porém, na base de dados SEL o campo *Tipo de Projeto* (se o projeto é de tempo real, científico, comercial, etc.), dentre outros importantes, não continha nenhuma informação e não pôde ser usado na implementação. Deve-se ter em mente o impacto que esta escolha e atribuição de pesos a estes atributos (ou características) têm no sentido de se obter uma boa ou má solução. Por isso, pode ser interessante se apoiar em um padrão que forneça recomendações.

Além disso, se a pessoa responsável por representar o conhecimento em uma Base de Casos não tiver uma experiência significativa no domínio de gerenciamento de projeto, é difícil identificar o conjunto de características importantes em uma busca e estabelecer quais são as características que devem ser consideradas prioritárias na

busca por casos similares. Nesse estudo de caso, foi possível perceber quão desafiador é o processo de estabelecer o conjunto de características que devem compor um caso. Esse desafio é percebido desde a representação até a análise dos resultados. Se tivesse utilizado um processo sistemático para identificar as métricas relevantes e um guia de gerenciamento de projetos que oferecesse recomendações sobre quais métricas utilizar, acredita-se que teria sido menos trabalhoso o passo referente à identificação do conjunto inicial de métricas, e os resultados da busca por casos similares teriam sido mais precisos.

Nesse estudo, foi feita uma análise de precisão da solução encontrada para um caso. Para tanto, foi tomado um dos casos contido na Base de Casos como sendo o novo caso, cuja estimativa de quantidade de horas de gerenciamento e quantidade de horas de programação deveriam ser estimadas. Consideraram-se dois momentos: 1) No primeiro momento, o caso permaneceu na base, e o programa foi executado, utilizando como dados de entrada do caso novo os dados daquele caso escolhido. Conforme esperado, o caso recuperado foi ele mesmo; 2) No segundo momento, o caso foi retirado da Base de Casos, e o programa foi executado, utilizando como dados de entrada para o caso novo os dados daquele caso escolhido. O caso obtido como resultado da busca, considerado pelo algoritmo de similaridade como sendo o mais semelhante, continha informações totalmente diferentes daquelas esperadas. Uma razão para essa imprecisão pode ser atribuída ao fato de não se ter utilizado um procedimento sistemático para a escolha das características que compõem os casos. Além disso, foi excluída uma grande quantidade de registros da Base de Experiência, que continha dados vazios para algumas das características que constituem os casos. Para solucionar esse último problema, pode-se utilizar alguma outra técnica de tratamento de dados perdidos

Referências bibliográficas

[1] Aamodt, A.; Plaza, E. **Case based reasoning: foundational issues, methodological variations, and systems approaches**. AI Communications, 7, 1994, p.39–59.

[2] Albrecht, A.J.; Gaffney, J.R. **Software function, source lines of code, and development effort prediction: a software science validation**. IEEE Transactions on Software Engineering 9(6), 1983, p. 639-648.

[3] Aggarwal, C.C., Parthasarathy, S. **Mining massively incomplete data sets by conceptual reconstruction.** Proceedings of the Seventh ACM SIGKDD Conference on Knowledge Discovery and Data Mining, , San Francisco, California, USA, 2001, p.227-232.

[4] Barcelos Tronto, I.F.; Sant'Anna, N.. **Um roteiro para construção de cubos e consultas OLAP.** In: IV Workshop dos Cursos de Computação Aplicada do INPE, São José dos Campos, 2004.

[5] Basili, V.R.; Caldiera, G.; Rombach, H.D. **Experience factory.** In J.J.Marciniak (ed.), Encyclopedia of Software Engineering, v.1, John Wiley & Sons, 1994a.

[6] Bisio, R.; Malabocchia, F. **Cost estimation of software projects through case base reasoning**. 1st Intl. Conf. on Case-Based Reasoning Research & Development,, Springer-Verlag, 1995, p.11-22.

[7] Boehn, B.W. **Software engineering economics.** Prentice-Hall, Englewood Cliffs, NJ, 1981.

[8] Finnie, G.R.; Wittig, G.E.; Desharnais, J.M. **Estimating software development effort with case-based reasoning.** 2nd Intl. Conf. on Case- Based Reasoning, 1997b, p.13-22.

[9] Kirsopp, C.; Shepperd, M.J.; Hart, J. **Search heuristics, case-based reasoning and software project effort prediction**. GECCO 2002: Genetic and Evolutionary Computation Conf., New York: AAAI, 2002.

[10] Mendonça, M.G..; Basili, V. R.**Validation of an approach for improving existing measurement frameworks,** IEEE Transactions on Software Engineering, vol.26, no 6, pp.484-500, June 2000.\

[11] Mendonça, M.G.; Sunderhaft, N.L. **Mining software engineering data: um survey.** A DACS State-of-the-Art Report, Data e Analysis Center for Software, SPO700-98-D-4000, 1999. At. www.dacs.dtic.mil/techs/datamining/datamining.pdf

[12] MSR, **Proceedings** 1st International Workshop on Mining Software Repositories, Edinburgh, Scotland, United Kingdom , 25 May, 2004.

[13] Paul, R.A.; Kunii, T. L.; Shinagawa, Y. ; Khan, M.F.**Software metrics knowledge and database for project management.** IEEE Transaction Knowledge Data Engineering, 11(1), 1999, p.255-264.

[14] Roth, P. **Missing Data: A Conceptual Review for Applied Psychologists**. Personnel Psychology, v.47, 1994, p.537-560.

[15] Sedigh-Ali, S.; Ghafoor, A.; Paul, R.A. **A metric-guided framework for cost and quality management of component-based software.** Lecture Notes in Computer Science, 2003, p.374-401.

[16] Sentas, P.; Angelis, L.; Stamelos, I.; Bleris, G. **Software productivity and effort prediction with ordinal regression.** Journal Information and Software Technology, 47, 2005, p.17-29.

[17] Troyanskaya, O.; Cantor, M.; Sherlock, G. **Missing value estimation methods for DNA microarrays.** Bioinformatics, Vol.17, 2001, p.520-525.

[18] Zhong, S.; Khoshgoftaar, T.M.; Seliya, N. **Analysing Software Measurement Data with Clustering Techniques.** IEEE Intelligent Systems, 2004, p.20-27.